Attorney's Docket No.: 10559/390001 / P10253

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: REDUCING LATENCY AND POWER IN ASYNCHRONOUS DATA TRANSFERS

APPLICANT: RICHARD P. MACKEY, DAVID R. SMITH AND JEFFREY J. MCCOSKEY

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. ET371085784US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

May 31, 2001
Date of Deposit

Signature

Gil Vargas
Typed or Printed Name of Person Signing Certificate

REDUCING LATENCY AND POWER IN ASYNCHRONOUS DATA TRANSFERS

TECHNICAL FIELD

This invention relates to transmitting data, and more particularly to reducing latency and power in asynchronous data transfers.

BACKGROUND

Transmitting data involves a transfer of data from a source domain to a destination domain. The transfer of data needs to be predictable to ensure that the destination domain receives the proper data from the source domain.

Meta-stability is the ability of a logic component (for example a flip-flop) to possess an indeterminate/unpredictable value for a finite period of time. Meta-stability occurs when the logic component attempts to capture (that is, sample) data before the data (that is, the information intended to occur on the component) is stable.

The probability of meta-stability directly effects the mean-time-between-failure ("MTBF") calculation for a product. As the MTBF increases, the usable duration and desirability of the product decreases. Thus, meta-stability needs to be avoided.

Asynchronous data transfers involve transmitting data from a source domain (for example, a microprocessor) to a

destination domain (for example, a memory device) operating (that is, capturing or sampling data) at different clock phases and/or frequencies. Asynchronous data transfers need an interface that compensates for the effects of meta-stability, ensuring the data sent between the two domains is stable before sampling.

Latency is a delay in the transfer of data that an interface introduces to compensate for the effects of meta-stability.

DESCRIPTION OF DRAWINGS

FIG. 1 is a prior art block diagram of a computer system.

FIG. 2 is a prior art block diagram of a two double flip-flop parallel synchronizer.

FIG. 3 is block diagram of a computer system according to one embodiment of the invention.

FIG. 4 is a block diagram of a domain-synchronizing controller according to one embodiment of the invention.

FIG. 4A is a block diagram of an alternative embodiment of FIG. 4.

FIG. 5 is a block diagram of an enable controller according to one embodiment of the invention.

FIG. 6 is a flow diagram according to one embodiment of the invention.

FIG. 7 is a block diagram a computer system according to one embodiment of the invention.

FIG. 8 is a block diagram of a gated-clock in FIG. 7.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

FIGS. 1 and 2 illustrate a prior art scheme that compensates for asynchronous data transfers between source and destination domains. In particular, system 100 includes source domain 102 operating at source clock SCLK connected to destination domain 104 operating at destination clock DCLK by interface 106. Interface 106 includes source register 111, a destination register 113 and a two double flip-flop parallel synchronizer 112.

Parallel synchronizer 112 (FIG. 2), which is also know as a 'fly-wheel', includes two sets of three flip-flops 202a-c and 204a-c and inverter 206 connected in a loop. The first set of three flip-flops 202a, 202b and 202c operate at the source clock SCLK phase and frequency. The second set of three flip-flops 204a, 204b and 204c operate at the destination clock DCLK phase and frequency.

Inverter 206 is positioned between the output Q and input D of two of the flip-flops in the loop and operates to invert

the output Q of one flip-flop before it reaches the input D of the next flip-flop. Inverter 206 operates to create a synchronous pulse signal propagating on wire 208 between the outputs Q and inputs D of flip-flops 202a-c and 204a-c.

The synchronous pulse signal propagates (that is, is sampled by the flip-flops) continuously through each flip-flop 202a-c at a rate dictated by the source clock SCLK and each flip-flop 204a-c at a rate dictated by the destination clock DCLK. The continuous synchronous pulse signal operates to produce a stable source enable signal 115 and a stable destination enable signal 117.

The stable source enable 115 and destination enable 117 signals operate to enable source register 111 and destination register 113 to accept data from their respective domains at a rate for which both of the domains may handle a transfer of data. In this system, the latency for transferring data between the source and destination domain is two or three clock cycles in source domain and two or three clock cycles in the destination domain for a total of up to six clock cycles for each data transfer. This system assumes that data is continuously transferred between two domains in a 1-to-1 ratio. This is not always the case.

A system 300 (FIG. 3) includes a source domain 302 and a destination domain 304 connected by an interface 306. Source

domain 302 (here for example, a microprocessor) operates at source clock SCLK phase and frequency. Destination domain 304 (here for example, a memory device) operates at destination clock DCLK phase and frequency.

Interface 306, includes a N-bit interface (for example, a N-bit bus), source registers 311, destination registers 313, domain-synchronizing controller 312 and enable controller 314. Here enable controller 314 includes source-enable controller 316 and destination enable controller 318.

Source registers 311, here two sets of N-enable flip-flops divided into Data Out and Data In portions, operate at clock signal SCLK and function to capture data according to N-source enable signals 315 from domain-synchronizing controller 312. Destination registers 313, here two sets of N-enable flip-flops divided into Data Out and Data In portions, operate at clock signal DCLK and function to capture data according to N-destination enable signals 317 from domain-synchronizing controller 312. Here, Data In for both source and destination registers 311 and 313 are connected to their respective domains through multiplexors 321 and 323 which may be included to provide for the sequential transfer of data into domains 302 and 304.

Multiplexor 321 provides data within the Data In portion of source registers 311 to source domain 302 according to

source input select signal 325 from source-enable controller 316. Multiplexor 323 provides data within the Data In portion of destination register 313 to destination domain 304 according to destination input select signal 327 from destination enable controller 318.

Domain-synchronizing controller 312 includes N-parallel synchronizers 400 (FIG. 4). Here, parallel synchronizer 400 includes flip-flops 402a-c and 404a-c and logic components 415, 416, 417, 418, 419, 420 and 421. Flip-flops 402a-c and 404a-c and logic components 416, 418, 419 and 420 are connected together in a loop which functions to produce a controllable synchronous-pulse signal Y (that is, a stoppable 'fly-wheel').

Logic component 419, here an XOR gate, is positioned between the output Q of flip-flop 402b and the input of logic component 416. Logic component 419 operates to invert the output Q of flip-flop 402b producing synchronous-pulse signal Y propagating through synchronizer 400.

Logic component 416, here a multiplexor, is positioned between the output of logic component 419 and the input D of flip-flop 402c. Logic component 416 operates to select either the output of logic component 419 or output Q of flip-flop 402c according to destination inhibit signal 331.

The selection of the output of logic component 419 allows synchronous-pulse signal Y to propagate through flip-flop 402c. The selection of output Q of 402c prevents the propagation of synchronous-pulse signal Y through flip-flop 402c. Accordingly, logic component 416 and flip-flop 402c, which together may be referred to as an enable flip-flop, introduce a control (that is, a stopping point in the synchronous-pulse signal Y) permitting destination inhibit signal 331 to control the propagation of signal Y through synchronizer 400.

Similarly logic component 418, here also a multiplexor, is positioned between the output of logic component 420, here an XOR gate and the input D of flip-flop 404a. Logic component 418 operates to select either the output of logic component 420, here operating as a buffer, or output Q of flip-flop 404a according to source inhibit signal 329.

The selection of the output of logic component 420 allows synchronous-pulse signal Y to propagate through flip-flop 404a. The selection of output Q of 404a prevents synchronous-pulse signal Y from propagating through flip-flop 404a. Accordingly, logic component 418 introduces a second control permitting source inhibit signal 329 to control the propagation of signal Y through synchronizer 400.

The controllable synchronous-pulse signal Y propagates through each flip-flop 402a-c at a rate dictated by the source clock SCLK and through each flip-flop 404a-c at a rate dictated by the destination clock DCLK. The stoppable 'fly-wheel' operates to produce a stable, synchronous source enable signal 315 and a stable, synchronous destination enable signal 317 when destination inhibit signal 331 and/or source inhibit signal 329 allow synchronous-pulse signal Y to propagate through controllable synchronizer 400. The generation of the source and destination inhibit signals 329 and 331 will be described in greater detail below.

Logic components 419 and 420 operate with inverter 421 to determine which side (that is, either the source or destination side) of synchronizer 400 to originate synchronous signal Y. Here, line 423 connects ground GND inverter 421 and XOR gate 420 causing XOR gate 420 to operate as a buffer and XOR gate 419 to operate as an inverter causing synchronous signal Y to originate on the source side of synchronizer 400. In an alternative embodiment, line 423 may be connected to Vdd causing XOR gate 419 to operate as a buffer and XOR gate 420 to operate as an inverter causing synchronous signal Y to originate on the destination side.

Source-enable signal 315 is produced by logic component 415, here an XOR gate positioned between the output Q of flip-

flop 402b and the output Q of flip-flop 402c. Accordingly, logic component 415 produces source-enable signal 315 for a full SCLK clock cycle when the output Q of flip-flop 402b is opposite the output Q of flip-flop 402c.

Similarly, destination-enable signal 317 is produced by logic component 417, here an XOR gate positioned between the output Q of flip-flop 404b and the output Q of flip-flop 404a. Accordingly, logic component 417 produces destination-enable signal 317 for a full DCLK clock cycle when the output Q of 404b is opposite the output Q of flip-flop 404a.

An alternative embodiment for producing source enable signal 315 is shown in FIG. 4a. Here, flip-flop 402d is introduced to drive source enable signal 315 and logic component 416. Accordingly, flip-flop 402d operates to reduce the load (for example, stray capacitance) on synchronous control signal Y propagating about synchronizer 400. A flip-flop (not shown) may also be similarly introduced to drive destination enable signal 317 and logic component 418 of flip-flop 404a.

Domain-synchronizing controller 312 includes N-parallel synchronizers 400 (that is, N-controllable 'fly-wheels') which produce N-stable source enable and destination enable signals 315 and 317, according to N-source inhibit and N-destination inhibit signals 329 and 331. The N-stable source enable and

destination enable signals 315 and 317 operate to enable N-source register 311 and N-destination register 313 to accept data at a rate which both domains may handle with reduced meta-stability.

Enable controller 314 (FIG. 3) here includes a source-enable controller 316 and a destination enable controller 318. Source-enable controller 316 here includes (FIG. 5) an event detector 502 and a counter 504. Similarly, destination enable controller 318 here also includes an event detector 506 and a counter 508. Source-enable controller produces N-source inhibit signals 329 and source input select signals 325. Destination enable controller 318 produces N-destination inhibit signal 331 and destination input select signals 327.

N-source inhibit signals 329 and N-destination inhibit signals 331 operate to reduce latency in the transfer of data between source domain 302 and destination domain 304 by tuning (that is, optimizing) the rate at which the synchronous-pulse signal propagates though each parallel synchronizer 400 in domain-synchronizing controller 312.

For example, in a system 300 having a two-to-one clock ratio between SCLK and DCLK, the higher speed SCLK domain may have information to transfer twice as often as the slower speed DCLK domain. Here, data may only be generated and/or received every other SCLK cycle in the DCLK domain.

Accordingly, to prevent wasting logic and to reduce latency source inhibit signal 329 may be configured (that is, controlled) to prevent the propagation of the synchronous-pulse signal every other SCLK clock pulse. Similarly, for other clock phase and/or frequency ratios (for example, 3-to-1, 4-to-1 and 5-to-1) a similar tuning or optimization based on a clock phase and/or frequency ratio may be accomplished using source and destination inhibit signals 329 and 331 to increase performance and reduce latency for data transfers between domains.

For even better performance, n-source inhibit signals 329 and n-destination inhibit signals 331 may prevent the synchronous-pulse signal from propagating through synchronizer 400 and producing source and destination enable signals until new data is available at either source domain 302 and/or destination domain 304. Here, source event signal 333 and/or destination event signal 335 may be used to detect the presence of new data at either domain 302 and/or 304. Absent an indication of new data, the source and/or destination inhibit signals will be sent to prevent the production of a source-enable and/or destination-enable signal until new data is available at either the source and/or destination domain. Accordingly, source and/or destination inhibit signals 329 and 331 will prevent the occurrence of a source and/or destination

enable signal 315 and/or 317 by stopping the propagation of the synchronous-pulse signal until new data is available for transfer at the source and/or destination domain or based on clock ratio described above, whichever is greater.

The source-enable controller 316 and destination enable controller 318 may also operate to produce a source input select signal 325 and destination input select signal 327. The source input select and destination input select signals 325 and 327 operate to select the most up to date synchronized input for the source and destination domains 302 and 304.

FIG. 6. shows process 60 for minimizing latency while transferring data between a source and destination domain. Process 60 produces (601) an enable signal based on a synchronous-pulse signal. Here, the enable signal enables a register to capture data from a domain operating at a source clock SCLK. Process 60 controls (603) the production of the enable signal with a inhibit signal. Here, the inhibit signal prevents the synchronous-pulse signal from producing the enable signal until the data is available for transmission.

FIG. 7 shows a system 700 having gated SCLK 702 and DCLK 704 for driving registers 311 and 313. Gated SCLK 702 includes flip-flop 802 (FIG.8) and logic components 804, 806, 808 and 810 and uses n-source-enable signals 315 and n-source-inhibit signals 329. Gated DCLK 704 includes similar

components and n-destination-enable signals 317 and n-destination-inhibit signals 331.

Logic component 804 (FIG. 8), here an inverter, is positioned between source-inhibit signal 329 and logic component 808. Logic component 804 operates to invert n-source enable signal 329.

Logic component 808, here an AND-gate, receives the inverted n-source inhibit signal 329 and n-source enable signal 315. Accordingly, when the inverted source-inhibit signal 329 is low (that is, when source enable controller 316 stops synchronizer 400) and/or when source-enable signal 315 is low (that is, that is when synchronous-pulse signal has been stopped) then the output of logic component 808 will also be low.

Flip-flop 802 samples the output of logic component 808 on the inverse of clock signal SCLK. Flip-flop 802 captures the output of logic component 808 using the inverse of SCLK to prevent logic component 810 from changing when SCLK is high (that is, flip flop 802 only samples data from logic component 808 in the second half of the clock pulse SCLK).

Logic component 810, here an AND-gate, receives output Q from flip-flop 802 and combines it with clock signal SCLK to create a gated SCLK clock signal, one clock SCLK pulse wide. Accordingly, logic component 810 only draws power from clock

pulse SCLK to drive register 311 when both a source-enable signal 315 and a source-inhibit signal 329 indicate that a register 311 may capture data.

Destination enable signal 317, destination-inhibit signal 331 and DCLK may also be combined with a circuit 80, replacing source enable signal 315, source-inhibit signal 329 and SCLK to create a gated DCLK clock. Accordingly, in such a circuit the logic component 810 only draws power from clock pulse DCLK to drive register 313 when both a destination-enable signal 317 and a destination-inhibit signal 331 indicate that a register 313 may capture data.

A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, the number of flip-flops in a loop may vary depending upon the meta-stability requirements of the system and the positioning of the logic components the domain-synchronizing controller may vary. Accordingly, other embodiments are within the scope of the following claims.